

On the Adversarial Robustness of Online Importance Sampling

Yotam Kenneth-Mordoch 

Weizmann Institute of Science, Rehovot, Israel

Shay Sapir 

Weizmann Institute of Science, Rehovot, Israel

Abstract

Online sampling algorithms, which irrevocably either keep or discard each stream element, have seen wide use in streaming due to their efficiency and simplicity. Braverman et al. [NeurIPS 2021] claimed that *online importance-sampling* algorithms, where elements are sampled proportionally to some notion of importance, succeed with high probability when their input stream is *adaptively chosen by an adversary*. Unfortunately, their results on importance sampling do not beat trivial bounds in many instances. Therefore, we reopen the question about the robustness of online importance sampling to adaptive inputs. This question was also addressed by Jiang, Peng and Weinstein [FOCS 2023] for the problem of ℓ_2 -subspace embedding.

We develop a unified framework for online importance sampling algorithms in adaptive streams. This framework offers two main advantages: first, it provides better bounds than prior work, and second, it unifies and simplifies the analysis of importance sampling algorithms across different problems. We then leverage the framework to provide algorithms for cut sparsification in hypergraphs and ℓ_p -subspace embeddings in adaptive streams whose space complexity nearly matches the oblivious case (non-adaptive).

2012 ACM Subject Classification Theory of computation → Sketching and sampling; Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Importance sampling, Adversarial robustness, Streaming algorithms, Coresets, Cut sparsification, Subspace embedding

Digital Object Identifier 10.4230/LIPIcs.ESA.2026.113

Related Version *Full Version*: <https://arxiv.org/abs/2507.02394>

Acknowledgements We would like to thank Samson Zhou for pointing out the relevance of [34] for online sampling, and Sandeep Silwal for helpful comments on improving the readability of this work.

1 Introduction

The streaming model of computation is a rich algorithmic area, and particularly useful for large-scale data analysis. A streaming algorithm is given its input as a sequence of items that can only be read sequentially, and is required to compute some global function of the data. The main measure of a streaming algorithm's efficiency is its *space complexity*, i.e., the amount of space it uses. This model is particularly useful for the analysis of massive datasets, where the input is too large for the storage available to the algorithm. This occurs naturally in many instances such as computing statistics of large databases, IoT measurements and network traffic logs. A more restricted variant of the streaming model is the *online* model, where the algorithm may only store a small number of items (along perhaps with some auxiliary data structures) and its decisions are irrevocable, i.e., once an item is stored it may never be deleted. While such online algorithms in general provide weaker guarantees, they are often simpler to analyze and implement. For further motivation, see e.g. [24, 15].



© Yotam Kenneth-Mordoch and Shay Sapir;
licensed under Creative Commons License CC-BY 4.0

34th Annual European Symposium on Algorithms (ESA 2026).

Editors: Philip Bille, Seth Pettie, and Sabine Storandt; Article No. 113; pp. 113:1–113:19

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Most of the streaming literature assumes that the input stream is fixed in advance by an *oblivious adversary*. However, these assumptions do not necessarily hold, and a recent line of work [12, 11, 5, 35, 50, 4, 33, 10, 17, 6, 7, 46, 51, 18] considers the more difficult setting where the stream may depend on previous algorithm outputs, modeled by an *adaptive adversary*. That is, the algorithm must output a correct response after processing each item; the adversary may then observe these responses and choose the next stream element. The immediate motivation is when the input is controlled by a malicious party, but another motivating scenario is when a user repeatedly queries and updates a database based on the answers to previous queries. A streaming algorithm is called *adversarially-robust* if it succeeds with high probability against any adaptive adversary.

This adaptivity introduces dependencies that break the analysis of most algorithms designed for oblivious streams. Furthermore, there are several demonstrated adversarial attacks against classical algorithms, notably against linear sketches [32, 11, 17, 1, 21, 22, 29, 30, 23, 2]. On the other hand, in some cases, adversarial-robustness is obtained with space complexity similar to the oblivious setting. For example, adversarially-robust algorithms for frequency moments in insertion-only streams require at most a poly-logarithmic overhead [11, 50, 33, 7].

In this paper, we study the problem of constructing succinct representations of the stream. This is a common preprocessing step in many algorithms where the data is too large to be processed in its entirety. We study two seemingly unrelated problems in this vein: hypergraph cut sparsification and subspace embedding, both of which have seen wide use in streaming algorithms and beyond. In both problems, the goal is to preserve some approximate property of the data while using as little storage as possible. For hypergraphs, a natural generalization of graphs where hyperedges connect any number of vertices, we wish to find a small hypergraph that preserves all cuts up to a multiplicative $(1 \pm \epsilon)$ factor [31, 40, 19, 20, 36]. Applications of cuts in hypergraphs include scientific computing on sparse matrices [8], and clustering and machine learning [53, 52]. In subspace embedding, the input is a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ and the goal is to find a smaller matrix $\tilde{\mathbf{A}} \in \mathbb{R}^{n' \times d}$ such that for every $x \in \mathbb{R}^d$ we have $\|\tilde{\mathbf{A}}x\| \in (1 \pm \epsilon)\|\mathbf{A}x\|$ [25, 41, 43, 49]. Subspace embeddings have many applications in numerical linear algebra, see the surveys [48, 42]. We solve both problems through the construction of coresets, weighted subsets of the original stream that preserve the desired property. One useful approach for coreset construction in oblivious streaming is (online) importance sampling; a weighted sampling technique where each stream element is assigned an importance and sampled with probability proportional to it (in the online setting, items are assigned probabilities as they arrive and are sampled irrevocably).

The study of adversarially robust sampling was initiated by Ben-Eliezer and Yogev [12], who showed that with respect to a universe $\mathcal{U}$ and a collection of sets $\mathcal{R} \subseteq 2^{\mathcal{U}}$, uniformly sampling $O(\epsilon^{-2} \log |\mathcal{R}|)$ elements is both sufficient and necessary for ϵ -approximating the density of elements in every $S \in \mathcal{R}$, (see also [5] for better bounds parametrized by the Littlestone’s dimension of the underlying set system). Another line of work studies the robustness of online importance sampling algorithms [16, 34]. These results assume some “condition” bound κ on the stream; for example, the ratio between the minimum and maximum cut in a graph. The analysis of [16] showed that using κ^2 space complexity overhead in comparison to oblivious algorithms results in adversarial-robustness. Unfortunately, in graphs, the ratio between the minimum and maximum cuts is always at least the ratio between the minimum degree and half the number of edges in the graph, which is $\Omega(n)$. Therefore, their algorithm requires storing the entire graph. An improved overhead for the case of ℓ_2 -subspace embedding was later provided in [34].

Our work provides two main improvements to the construction of adversarially-robust online importance sampling algorithms. First, we provide improved bounds, and second, we provide a generic approach that unifies the analysis of importance sampling across many problems. (In contrast to prior work, where each problem required its own problem-specific analysis.) This unified approach simplifies the analysis and helps shed some light on the difficulty of obtaining better bounds for online sampling algorithms.

Finally, by combining our online algorithms with a technique of [27, 26], which integrates online sampling with the well-known merge-and-reduce framework, we obtain nearly-optimal hypergraph cut sparsification and ℓ_p subspace embedding.

Parallel work. The first version of this paper included only results on online sampling. Subsequently, the online posting of [26] inspired the addition of the technique combining online sampling with merge-and-reduce to further improve space complexity. We note that our improved online-sampling algorithms are required for obtaining adversarially-robust streaming algorithms that match their oblivious counterparts.

1.1 Hypergraph Cut Sparsification

A hypergraph $G = (V, E)$ is a generalization of a graph, where edges (called hyperedges) can connect any number of vertices (i.e., every $e \in E$ is a subset of V). One fundamental object in the study of hypergraphs is a cut, which is a partition of the vertex set V into two disjoint sets $S \subseteq V$ and $V \setminus S$, and whose value is defined as $\text{cut}_G(S) := \sum_{e \in E} \mathbb{1}_{\{0 < |e \cap S| < |e|\}} \cdot w_e$. Notably, the number of hyperedges can be as large as $2^{|V|}$, and therefore, computing exact cuts in hypergraphs is often infeasible. This motivates the constructions of succinct cut sparsifiers that preserve the cut values of the hypergraph.

► **Definition 1.** *Given a hypergraph $G = (V, E)$, a reweighted subgraph $G' = (V, E')$ of G is called a quality $(1 \pm \epsilon)$ -cut sparsifier of G if,*

$$\forall S \subseteq V, \quad \text{cut}_{G'}(S) \in (1 \pm \epsilon) \cdot \text{cut}_G(S).$$

Hypergraph cut sparsifier construction is a well-studied problem [40, 19, 9, 20, 36, 44], including in the streaming setting [31, 45, 38, 37, 39]. We consider this problem in the insertion-only streaming model, where the hyperedges are given one at a time, and the stream's length is the number of edges denoted by m . Our first result is an adversarially-robust algorithm for hypergraph cut sparsification. Throughout we use $\tilde{O}(x)$ to hide polylogarithmic factors of x .

► **Theorem 2.** *Let $\epsilon > 0$ and a vertex set V of size n . There exists an algorithm that, given an adaptive stream of m hyperedges $e_1, \dots, e_m$ on V , maintains a $(1 \pm \epsilon)$ -cut sparsifier of $G_t = (V, \{e_i\}_{i=1}^t)$ for all $t \in [m]$. The algorithm succeeds with probability at least $1 - 1/\text{poly}(n)$ and stores at most $\tilde{O}(\epsilon^{-2}n)$ hyperedges.*

Our algorithm matches (up to polylog factors in $n, \log m, \epsilon^{-1}$) existing offline algorithms for hypergraph sparsification [40, 19, 20, 44]. It similarly matches existing algorithms for insertion-only (non-robust) streams [31, 45, 38, 37, 39]. Previously, [16] obtained adversarial-robust algorithms via the merge-and-reduce framework, which is a general technique that applies to coresets in general. For hypergraphs, this yields a robust sparsifier with $\tilde{O}(\epsilon^{-2}n \log^3(m/n))$ hyperedges. Note that since $\log m$ can be as large as n in hypergraphs, Theorem 2 offers substantial savings over existing adversarially-robust algorithms in the natural case when m is large. In addition, an adversarially robust hypergraph sparsification algorithm that

stores $\tilde{O}(\epsilon^{-2}n)$ hyperedges was proposed in [26]. However, their algorithm stores an auxiliary data-structure to compute sampling probabilities, which takes $\tilde{O}(\epsilon^{-2}n \text{poly}(r))$ storage, where r is the cardinality of the largest hyperedge in H . Noting that r can be as large as $\Omega(n)$ we find that the storage complexity of their algorithm is polynomially worse than ours.

Note that while the theorem is stated for unweighted hypergraphs, it can easily be extended to weighted hypergraphs by simulating the insertion of each hyperedge e with weight w_e as the insertion of w_e copies of e . This increases the storage requirement to $\tilde{O}(\epsilon^{-2}n \log \log W)$ hyperedges, where $W = \sum_{e \in E} w_e$ is the sum of all hyperedge weights. We also give an improved algorithm for the online setting, where items are sampled irrevocably as they arrive. Note that this algorithm has higher probability of success than Theorem 2.

► **Theorem 3.** *Let $\epsilon > 0$ and a vertex set V of size n . There exists an online sampling algorithm that, given an adaptive stream of m hyperedges $e_1, \dots, e_m$ on V , maintains a $(1 \pm \epsilon)$ -cut sparsifier of $G_t = (V, \{e_i\}_{i=1}^t)$ for all $t \in [m]$. The algorithm succeeds with probability at least $1 - 2^{-n}$ and stores at most $\tilde{O}(\epsilon^{-2}n^2 \log m)$ hyperedges.*

Previously, an adversarially-robust online sampling algorithm with $\tilde{O}(\kappa^2 \epsilon^{-2}n^2 \log m)$ edges for graphs was given in [16], where κ is the ratio between the smallest and largest cut in the graph. This result can be extended to hypergraphs using the bound on the number of hyperedges employed in the proof of Theorem 3. Unfortunately, κ can be $\Omega(2^n)$ and hence the algorithm does not improve upon the trivial solution of storing the entire stream in the worst case. In addition, an adversarially-robust online sampling algorithm storing $\tilde{O}(\epsilon^{-2}n^2 \log^2 m)$ hyperedges can be obtained using the techniques of [26], this algorithm has an $O(\log m)$ overhead factor in comparison to Theorem 3. Finally, note that in the online setting, there exists a lower bound of $\Omega(\epsilon^{-2}n \log m)$ on the number of hyperedges that must be stored in the construction of a cut sparsifier [37]. Hence, the gap between our result and the best possible for online sampling (even in non-adaptive streams) is $\Theta(n)$.

1.2 ℓ_p Subspace Embedding

We also consider a fundamental problem in numerical linear algebra, ℓ_p subspace embedding for fixed $p > 0$. In this problem, the input is a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ where $n \gg d$ and an accuracy parameter $\epsilon > 0$, and the goal is to produce a (smaller) matrix $\tilde{\mathbf{A}} \in \mathbb{R}^{n' \times d}$ such that $\|\tilde{\mathbf{A}}x\|_p^p \in (1 \pm \epsilon)\|\mathbf{A}x\|_p^p$ for all $x \in \mathbb{R}^d$, where $\|y\|_p^p = \sum_{i=1}^n |y_i|^p$ for $y \in \mathbb{R}^n$. A notable special case is $p = 2$, also known as spectral approximation. Oftentimes, it is desired that the rows of $\tilde{\mathbf{A}}$ are a (weighted) subset of the rows of $\mathbf{A}$, e.g., if the rows of $\mathbf{A}$ are sparse then so are the rows of $\tilde{\mathbf{A}}$. Therefore, we restrict the output matrix $\tilde{\mathbf{A}}$ to be constructed by a weighted subset of the rows of $\mathbf{A}$. In this setting, there are offline algorithms storing $\tilde{O}(\epsilon^{-2}d^{\max(1, p/2)})$ rows [25, 43, 49].

We consider the row-order streaming model, where the matrix is given row by row. Denote by $\mathbf{A}_i$ the matrix $\mathbf{A}$ restricted to the first i rows. Define the *online condition number* κ^{OL} of $\mathbf{A}$ to be the ratio between the largest singular value of the final matrix $\mathbf{A}_n \equiv \mathbf{A}$ and the smallest non-zero singular value across all intermediate matrices $\mathbf{A}_i$. We make the standard assumption that the entries of the matrix are integers bounded by $\text{poly}(n)$ (so they can be stored in memory using $O(\log n)$ bits).

► **Theorem 4.** *Fix constant $p > 0$, and let $\epsilon > 0$ and $d \in \mathbb{N}$. There exists an algorithm that, given an adaptive stream of rows $a_1, \dots, a_n \in \mathbb{R}^d$ whose entries are integers in $[-\text{poly}(n), \text{poly}(n)]$, maintains a $(1 + \epsilon)$ -approximate ℓ_p subspace embedding of $\mathbf{A}_t = [a_1; \dots; a_t]$ for all $t \in [n]$. The algorithm succeeds with high probability and stores $\tilde{O}\left(\epsilon^{-2}d^{\max(1, p/2)}(\log \log(n\kappa^{OL}))^4\right)$ rows.*

This algorithm matches existing non-robust streaming and offline algorithms up to $\text{poly}(\log d \cdot \log \log(n\kappa))$ factors [25, 43, 49, 24, 15]. Previously, there was an adversarially robust ℓ_p -subspace embedding with an overhead of $d\kappa^{OL}$ factor [16]. This bound was improved for $p = 2$ to $d \log \kappa^{OL}$ by [34]. Another approach by [16] uses the merge-and-reduce framework, and has an overhead of $O(\log^3 n)$ over offline constructions. Theorem 4 beats merge-and-reduce in the natural regime where n grows superpolynomially in d . Similarly to hypergraph sparsification, we also provide an online sampling algorithm.

► **Theorem 5.** *Fix constant $p > 0$, and let $\epsilon, \delta > 0$ and $d \in \mathbb{N}$. There exists an online algorithm that, given an adaptive stream of rows $a_1, \dots, a_n \in \mathbb{R}^d$ whose entries are integers in $[-\text{poly}(n), \text{poly}(n)]$, maintains a $(1 + \epsilon)$ -approximate ℓ_p subspace embedding of $\mathbf{A}_t = [a_1; \dots; a_t]$ for all $t \in [m]$. The algorithm succeeds with probability $1 - \delta$ and stores at most $O\left(\epsilon^{-2}(d \log \frac{n\kappa^{OL}}{\epsilon} + \log \frac{1}{\delta}) \cdot (d \log(n\kappa^{OL}))^{\max(1, p/2)}\right)$ rows for $p \neq 2$ and $O\left(\epsilon^{-2}(d \log \frac{n\kappa^{OL}}{\epsilon} + \log \log n + \log \frac{1}{\delta}) \cdot d \log(n\kappa^{OL})\right)$ rows for $p = 2$.*

This result matches the current best-known adversarially-robust *online* algorithms in row-order streams for $p = 2$ [34]. We extend it to all $p > 0$, which is straightforward given our framework. There remains a gap of roughly $O(d \log(n\kappa^{OL}))$ to the known online algorithms in the non-adaptive setting (suppressing logarithmic factors), which store $\tilde{O}(\epsilon^{-2}(d \log(n\kappa^{OL}))^{\max(1, p/2)})$ rows [49].

1.3 Organization

The rest of the paper is organized as follows. Section 2 provides an overview for the techniques used in the proofs, then in Section 3 we provide the proofs for our self-weighted online sampling framework. Finally, in Appendix A we give the details for hypergraph cut sparsification. The proof for subspace embedding is given in the full version of the paper.

2 Technical Overview

Our adversarially-robust algorithms are built upon the following scheme. We first give a framework for adversarially-robust online importance sampling algorithms that choose sampling probabilities based solely on items sampled so far, which we term *self-weighted*. We then combine these algorithms with the widely applicable merge-and-reduce framework.

Our online sampling result provides a clean and generic approach for adversarially-robust self-weighted sampling algorithms by unifying the approach of [16, 34] to make it easily applicable to any self-weighted online sampling algorithm. Furthermore, our approach improves the parameters of their constructions. The proof is based on one-dimensional importance sampling, which we show is inherently adversarially robust. We extend this result by using a union bound on all “dimensions”, as we discuss at the end of the next section.

Our combination of adversarially-robust self-weighted online sampling with the merge-and-reduce framework is formalized as a black-box wrapper. This technique was used before to improve the storage complexity of *oblivious* streams [27, 26]. (Which [26] use as a basis for an adversarially-robust algorithm using the computational-paths framework of [11].) The basic idea is that the output sequence of an online algorithm can be fed, without storing it, as a virtual input stream to a merge-and-reduce algorithm. The sampling probabilities for the online sampling algorithm are then obtained based on the output of the merge-and-reduce algorithm. The correctness argument for adversarial-robustness is a bit delicate, but it essentially follows from the adversarial-robustness of merge-and-reduce [16], and the fact that our online sampling algorithms are self-weighted.

2.1 Self-Weighted Online Sampling Framework

Our framework is based on the one-dimensional case, where one wishes to estimate the sum of a stream of non-negative numbers using importance sampling. Intuitively, the process involves sampling each number with probability proportional to its contribution to the sum of all numbers seen so far. Denoting the stream by $x_1, \dots, x_m \in \mathbb{R}_+$, at time t , we sample x_t with probability $a \frac{x_t}{\sum_{i=1}^t x_i}$, where a is an *amplification parameter* that increases sampling probabilities to ensure correctness. This process is formalized in the following definition.

► **Definition 6** (One Dimensional Self-Weighted Online Importance Sampling). *Given an input stream $x_1, \dots, x_m \in \mathbb{R}_+$, self-weighted online importance-sampling with amplification parameter $a > 1$ is the following algorithm. Upon receiving item x_t , set $1 \geq p_t \geq \min\{1, a \frac{x_t}{x_t + \sum_{i=1}^{t-1} \tilde{x}_i}\}$, and use fresh randomness to compute*

$$\tilde{x}_t = \begin{cases} \frac{x_t}{p_t} & \text{w.p. } p_t, \\ 0 & \text{otherwise.} \end{cases}$$

For every $t \leq m$, return $\sum_{i=1}^t \tilde{x}_i$ as an estimate for $\sum_{i=1}^t x_i$.

We say that $\tilde{y}$ is a $(1 + \epsilon)$ -approximation of y if $\tilde{y} \in (1 \pm \epsilon) \cdot y$. The adversarial robustness of importance sampling was first examined in [16], who showed that given a deterministic (but crude) bound $\Delta > 1$ on the input, which is roughly the sum of elements in the stream in the worst-case dimension, one can get an adversarially-robustness by paying a $\text{poly}(\Delta)$ factor in the storage complexity compared to the non-adaptive setting. This result was improved for ℓ_2 subspace embedding, to a factor of roughly $\log \Delta$ [34].

Our approach extends the techniques of [34] to all self-weighted online sampling problems. Additionally, using an easy observation, we improve the “cost” of adversarial robustness from $\log \Delta$ to $\log \log \Delta$.¹ Finally, this framework is widely applicable, as is demonstrated by our two applications, hypergraph cut sparsification and subspace embedding.

► **Theorem 7** (Adversarially-Robust Self-Weighted Importance Sampling (Correctness)). *Let $\epsilon, \delta \in (0, 1)$, $\Delta > 1$. Given an adaptive stream of non-negative numbers $x_1, \dots, x_m \in \mathbb{R}_+$ such that $\frac{\sum_{i=1}^m x_i}{x_1} \leq \Delta$; with probability at least $1 - \delta$, self-weighted online importance-sampling with amplification parameter $a = O(\epsilon^{-2} \log \frac{\log \Delta}{\epsilon \delta})$ returns a $(1 + \epsilon)$ -approximation of $\sum_{i=1}^t x_i$ for all $t \in [m]$.*

Note that our theorem focuses on bounding the amplification parameter a and not the actual sample size. We note that factor $\log \log \Delta$ seems necessary also for algorithms in oblivious streams that guarantee correctness at every time step.

The assumption $\frac{\sum_{i=1}^m x_i}{x_1} \leq \Delta$ can be replaced with the natural (and stronger) assumption that the updates are bounded in $[1, \Delta']$, which yields $\Delta \leq m\Delta'$. Moreover, some bound on update size must be assumed, since otherwise, the sum of online importances $\sum_{t=1}^m \frac{x_t}{x_t + \sum_{i=1}^{t-1} \tilde{x}_i}$ may be as large as $\Omega(m)$, and the algorithm must then store the entire stream. For example, consider the stream $1, 2, 4, \dots, 2^m$ with amplification parameter $a = O(1)$. At time $t \in [m]$, we have $\frac{2^t}{\sum_{i=0}^t 2^i} = \Omega(1)$, hence $p_t = 1$, and eventually all items are sampled. We now give an overview of the proof of Theorem 7.

¹ Ignoring factors depending on ϵ^{-1} .

The adversary's power. Recall that every item is irrevocably kept with probability proportional to its importance at the moment it arrives. Therefore, once an item is processed by the algorithm, the adversary cannot affect it anymore. Hence, the adversary can only hope to “fail” the algorithm by either changing the sampling probabilities or by adding “bad” items to the stream.

Our proof follows by separating the adversary's power into these parts: inserting items and setting sampling probabilities. We first show that if the sampling probabilities are “good”, then the algorithm maintains an accurate estimate with high probability (for the amplification parameter a of Theorem 7). We then show through a bootstrapping argument that the sampling probabilities are indeed “good” with high probability.

Sampling game. For the first part, consider a two-player game between a sampling algorithm, SamplingAlg, and an adversary Adversary. In this game, the adversary essentially has more power compared to Theorem 7 – the adversary also picks the sampling probabilities subject to some constraint. The game is as follows. Let $\epsilon \in (0, 1)$. First, SamplingAlg picks a number $a \geq 1$. Then the game proceeds in rounds, where in the t -th round,

1. Adversary picks a number $x_t > 0$, assigns it a sampling probability $\min\{a \frac{x_t}{\sum_{i=1}^t x_i}, 1\} \leq p_t \leq 1$, and sends (x_t, p_t) to SamplingAlg.
2. SamplingAlg uses fresh randomness, computes

$$\tilde{x}_t = \begin{cases} \frac{x_t}{p_t} & \text{w.p. } p_t, \\ 0 & \text{otherwise,} \end{cases}$$

and sends $\tilde{x}_t$ to Adversary.

The goal of SamplingAlg is to maintain $\sum_{i=1}^t \tilde{x}_i \in (1 \pm \epsilon) \sum_{i=1}^t x_i$ for all t , and the goal of Adversary is to cause SamplingAlg to return an incorrect estimate at some time t . Notice that this game is similar to Definition 6, but now the sampling probabilities p_t are constrained by the exact quantity $\sum_{i=1}^t x_i$, rather than its approximation $x_t + \sum_{i=1}^{t-1} \tilde{x}_i$. The following technical lemma states that for the amplification parameter a of Theorem 7, Adversary loses the game with high probability.

► **Lemma 8 (Sampling Game).** *Let $\Delta > 1, \epsilon, \delta \in (0, 1)$. Consider the game between Adversary and SamplingAlg with the restriction that $\frac{\sum_{i=1}^m x_i}{x_1} \leq \Delta$. For a suitable $a = O(\epsilon^{-2} \log \frac{\log \Delta}{\epsilon \delta})$, SamplingAlg wins the game with probability $1 - \delta$.*

In the oblivious (non-adaptive) setting, one can prove a similar lemma, essentially by a Bernstein's bound and by observing that the variance of $\sum_{i=1}^t x_i - \tilde{x}_i$ is bounded by $\frac{1}{a} (\sum_{i=1}^t x_i)^2$. One might wish to use a similar method for the one-dimensional case in the adaptive setting, by defining an appropriate martingale sequence $X_t = \sum_{i=1}^t x_i - \tilde{x}_i$, and applying Freedman's inequality (which is analogous to Bernstein's inequality). Unfortunately, in order to apply Freedman's inequality, we need a bound on $\sum_{i=1}^t x_i$, which is a random variable in the adaptive setting. We overcome this challenge by partitioning the stream into $O(\epsilon^{-1} \log \Delta)$ phases, based on rounding $\sum_{i=1}^t x_i$ to the nearest power of $(1 + \epsilon)$. Note that this partition is used only for the analysis.

For each phase k , we create a virtual stream of items $\{x'_i\}_{i \in [m]}$, such that $x'_j = x_j$ for all $j \in [m]$ such that $\sum_{i=1}^j x_i \leq (1 + \epsilon)^k \cdot x_1$, and otherwise $x'_j = 0$. This yields a deterministic bound of $\sum_{i=1}^t x_i \leq (1 + \epsilon)^k \cdot x_1$ in the k -th stream. We then define an appropriate martingale sequence for each virtual stream, and use this deterministic bound on $\sum_{i=1}^t x_i$ to bound the martingale sequence using Freedman's inequality. The proof is concluded by applying a union bound over all virtual streams.

Note that previous work employed a similar technique, however it was only applied to the problem of ℓ_2 subspace embedding, and furthermore it used $O(\epsilon^{-1}\Delta)$ phases, instead of $O(\epsilon^{-1}\log \Delta)$ phases as in our case [34]. Thus, they require the amplification parameter a to be $O(\log(\epsilon^{-1}\Delta))$ compared to the $O(\log(\epsilon^{-1}\log \Delta))$ that we achieve. Finally, our proof technique has two main advantages. First, by abstracting the problem to a game we obtain a simpler proof, which is easier to follow and extend. Second, it enables the application of the same technique to other problems. Therefore, we hope that this presentation will be a useful basis for future works. For further details, see Section 3.

Bootstrapping the sampling probabilities. We now explain how to strengthen Lemma 8 to the case when the sampling probabilities are not computed deterministically, thus proving Theorem 7. This follows by formalizing online importance sampling, as a version of the game between Adversary and SamplingAlg. In this version, $a = O(\epsilon^{-2} \log \frac{\log(\Delta)}{\epsilon\delta})$ as in Lemma 8, and Adversary is required to choose $p_t = \min \left\{ \frac{2ax_t}{x_t + \sum_{i=1}^{t-1} \tilde{x}_i}, 1 \right\}$ (i.e., the “online” importance of x_t) whenever it is a valid strategy. When this strategy is not valid, the adversary is not restricted. Notice that if SamplingAlg’s output was correct up to time t , then the above is a valid strategy for the game, i.e., if $\sum_{i=1}^{t-1} \tilde{x}_i \in (1 \pm \epsilon) \sum_{i=1}^{t-1} x_i$, then

$$\frac{2ax_t}{x_t + \sum_{i=1}^{t-1} \tilde{x}_i} \geq \frac{2ax_t}{x_t + (1 + \epsilon) \sum_{i=1}^{t-1} x_i} \geq \frac{ax_t}{\sum_{i=1}^{t-1} x_i}, \quad (1)$$

and the strategy is valid.

Proof of Theorem 7. We consider a dominant strategy for an adversary that tries to fool online importance sampling. For every $t \in [m]$, the adversary picks some χ_t of their choice that satisfies $(\chi_t + \sum_{i \leq t} x_i)/x_1 \leq \Delta$, which can depend on past randomness. If $\frac{2a\chi_t}{\chi_t + \sum_{i=1}^{t-1} \tilde{x}_i} \geq \frac{a\chi_t}{\chi_t + \sum_{i=1}^{t-1} x_i}$, the adversary chooses $x_t = \chi_t$, and otherwise, they choose $x_t = 0$. This is a dominant strategy, since the adversary can choose a strategy freely while $\sum_{i=1}^t \tilde{x}_i \in (1 \pm \epsilon) \sum_{i=1}^t x_i$, and when this condition is violated, the future choices of the adversary do not affect the outcome (adversary had already won).

Additionally, the strategy described above, along with the “online importance” of χ_t , $p_t = \min \left\{ \frac{2a\chi_t}{\chi_t + \sum_{i=1}^{t-1} \tilde{x}_i}, 1 \right\}$, is a valid strategy for Lemma 8. (The factor 2 can be incorporated in the parameter a .) Therefore, such an adversary loses with probability at least $1 - \delta$, and since their strategy is dominant, this concludes the proof. ◀

Beyond the one-dimensional case. To obtain our bounds for hypergraph sparsification and subspace embedding, we apply Theorem 7 with a union bound that we call “uniform”. Before going into specific details, we define a general setting for which our approach is applicable, and is captured by the notion of coresets. We consider problems defined by a universe $\mathcal{U}$ (e.g. $\mathcal{U} = \mathbb{R}^d$ for points, or $\mathcal{U} = 2^{[n]}$ for hyperedges), query set $\mathcal{Q}$ (e.g. $\mathbb{R}^d$ for subspace embedding, or cuts for hypergraphs), and cost function $c : \mathcal{U} \times \mathcal{Q} \rightarrow \mathbb{R}_+$, where $\mathbb{R}_+$ is the set of all non-negative reals. For every query $x \in \mathcal{Q}$ and weight vector $w \in \mathbb{R}_+^{|\mathcal{U}|}$, where $\mathbb{R}_+^{|\mathcal{U}|}$ is the set of all vectors in $\mathbb{R}^{|\mathcal{U}|}$ whose entries are non-negative, we define its cost in regards to x as

$$C(w, x) := \sum_{u \in \mathcal{U}} w_u \cdot c(u, x).$$

Denote the stream as $\{(u_i, a_i)\}_{i=1}^m$ where $u_i \in \mathcal{U}$ is an element and $a_i \in \mathbb{R}_+$ is its weight. Define $\mu_i := a_i \cdot e_{u_i}$ where e_{u_i} is a unit vector in the direction u_i . Using this, we represent the input stream as $(\mu_1, \dots, \mu_m)$, and define the vector of all inputs up to time t by $w_t = \sum_{i \leq t} \mu_i$. (For example, think of w_t as a weighted hypergraph in a vector form.) Similarly, denote the output stream of the algorithm by w'_t . We say the stream is adaptive if for every $t \leq m-1$, μ_t may depend on $\{w'_1, \dots, w'_{t-1}\}$. A coreset is defined as follows.

► **Definition 9 (Coreset).** For a cost function C , a $(1 + \epsilon)$ -coreset of w is a vector $w' \in \mathbb{R}_+^{\mathcal{U}}$ such that $\text{supp}(w') \subseteq \text{supp}(w)$ and

$$\forall x \in \mathcal{Q}, \quad C(w, x) \in (1 \pm \epsilon) \cdot C(w', x).$$

The size of a coreset w' is the number of non-zero coordinates it has. Coresets also satisfy the following properties.

- **Reduce:** If w' is an $(1 + \epsilon_1)$ -coreset of w and w'' is an $(1 + \epsilon_2)$ -coreset of w' , then w'' is an $(1 + \epsilon_1)(1 + \epsilon_2)$ -coreset of w .
- **Merge:** If w'_1 is an $(1 + \epsilon_1)$ -coreset of w_1 and w'_2 is an $(1 + \epsilon_2)$ -coreset of w_2 , then $w'_1 + w'_2$ is a $(1 + \max\{\epsilon_1, \epsilon_2\})$ -coreset of $w_1 + w_2$.

► **Remark 10.** Note that both hypergraph cut sparsifiers and ℓ_p subspace embeddings are in fact coresets, this is formally proven in Appendix A.2 and in the full version of the paper respectively.

We say that a streaming algorithm computes/maintains a coreset if for every $t \leq m$, it outputs a vector $w'_t \in \mathbb{R}_+^{\mathcal{U}}$ that is a coreset of w_t . Our framework crucially builds on online sampling algorithms, see e.g. [3, 24, 45, 37], and we use a restricted notion that we call *self-weighted*, as follows.

► **Definition 11 (Self-Weighted Online Importance Sampling).** Given a stream $\mu_1, \dots, \mu_m$, self-weighted online importance sampling with amplification parameter $a > 1$ is the following algorithm. Upon receiving item μ_t , set $1 \geq p_t \geq \min\{1, a \cdot \max_{x \in \mathcal{Q}} \frac{C(\mu_t, x)}{C(w'_{t-1} + \mu_t, x)}\}$ and use fresh randomness to compute

$$\mu'_t \leftarrow \begin{cases} \frac{\mu_t}{p_t} & \text{with probability } p_t, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Maintain $w'_t = \sum_{i=1}^t \mu'_i$.

Putting It All Together. In our applications, we compute a coreset as follows. Assume we are given some net $\mathcal{Q}' \subseteq \mathcal{Q}$ of bounded size, such that if $C(w', x) \in (1 \pm \epsilon)C(w, x)$ for every $x \in \mathcal{Q}'$, then $C(w', y) \in (1 \pm O(\epsilon))C(w, y)$ for every $y \in \mathcal{Q}$. Fix some $x \in \mathcal{Q}'$ and observe that $C(w, x)$ is exactly a sum of elements as described in Theorem 7. If we sample the element at time t with probability at least $p_t = a \frac{C(\mu_t, x)}{C(w'_t + \mu_t, x)}$ (for brevity, we omit the minimum with 1), then by Theorem 7, we obtain a $(1 + \epsilon)$ -approximation of $C(w, x)$, for an appropriate $a > 1$.

To approximate $C(w, x)$ for all $x \in \mathcal{Q}'$, we sample the element at time t with probability $a \cdot \max_{x \in \mathcal{Q}'} \frac{C(\mu_t, x)}{C(w'_t + \mu_t, x)}$. Next, we amplify the success probability of Theorem 7 by a $\frac{1}{|\mathcal{Q}'|^{1/2}}$ (which increases space by a $\log |\mathcal{Q}'|$ factor). By a union bound, we obtain correctness for all $x \in \mathcal{Q}'$, and hence w'_t is a coreset of w_t . We call such a union bound “uniform” because it is based on the same net for all $w \in \mathbb{R}_+^{\mathcal{U}}$.

To make this idea concrete, we now give an overview of the construction of a cut sparsifier for hypergraphs (Theorem 3). The details for ℓ_p subspace embedding are similar and omitted for brevity. Throughout, let $G = (V, E)$ be some hypergraph. To construct a cut sparsifier of

G , we choose the net $\mathcal{Q}'$ to be the entire set of cuts $2^{[n]}$. It is clear that $C(w, x)$ is simply the number of hyperedges crossing the cut for every $x \in 2^{[n]}$. Therefore, the sampling probability of each hyperedge is given by the smallest cut which intersects it, and then applying a union bound on all the cuts to conclude the proof.² The bound on the size of the sparsifier follows from structural analysis akin to [3] and is detailed in Appendix A.

Gap to Oblivious Online Sampling. Unfortunately, the “uniform” union bound approach leaves a sizable gap to the oblivious setting. For example, for ℓ_p subspace embeddings, in the oblivious setting, one can directly analyze the supremum of a certain quantity over the set $\{x : \|\mathbf{A}x\|_p = 1\}$ using a standard symmetrization argument and some other clever arguments. In comparison, our approach requires a uniform high probability bound for each element in the net $\mathcal{Q}$. It is unclear how to employ such symmetrization arguments in the adaptive setting, since the set $\{x : \|\mathbf{A}x\|_p = 1\}$ is now a random variable. Closing the gap between Theorems 3 and 5 and the oblivious setting for online algorithms remains open.

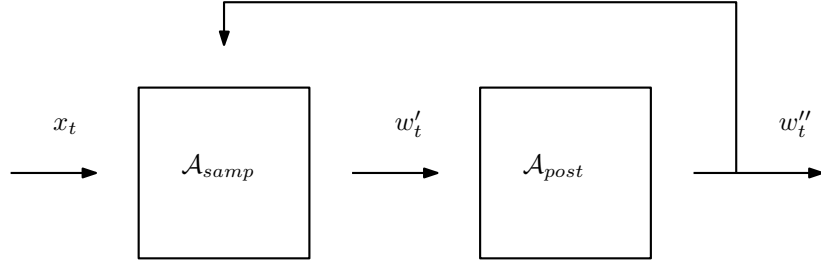
2.2 Black-Box Wrapper: Online Sampling and Merge-and-Reduce

Our second contribution is a black-box wrapper, inspired by [27, 26], that takes a self-weighted online sampling algorithm $\mathcal{A}_{\text{samp}}$ and combines it with a merge-and-reduce algorithm to produce a streaming algorithm with smaller space complexity (though it is no longer an online algorithm). At a high level, the merge-and-reduce framework maintains a coreset of its input stream by recursively applying an offline coreset construction algorithm. The basic idea of [27, 26], which they use in oblivious streams, is that the output stream of an online algorithm $\mathcal{A}_{\text{samp}}$ can be fed as a virtual input stream, without storing it, into a merge-and-reduce algorithm, $\mathcal{A}_{\text{post}}$. When a new item arrives, $\mathcal{A}_{\text{samp}}$ computes its sampling probability based on the current coreset maintained by $\mathcal{A}_{\text{post}}$. See Figure 1 for an illustration of the wrapper. Our wrapper can yield improved storage complexity compared to directly using merge-and-reduce, since the storage complexity of merge-and-reduce algorithms depends on the stream length.

It is known that the merge-and-reduce framework is adversarially-robust [16] (we provide a proof of this fact in the full version of the paper). We use this fact to show that if $\mathcal{A}_{\text{samp}}$ is adversarially-robust, then so is the output of the combined algorithm. Therefore, our wrapper yields a framework for obtaining adversarially-robust algorithms, by combining any adversarially-robust self-weighted online sampling algorithm and offline coreset construction algorithm.

Existing work used a different approach to obtain adversarially-robust algorithms by combining online sampling and merge-and-reduce [26]. First, construct a non-robust algorithm whose space grows as $\text{poly}(\log \log \frac{1}{\delta})$, where δ is the failure probability. Adversarial-robustness is obtained using the computational paths framework of [11] to apply a union bound over an exponential number of possible executions to obtain adversarial-robustness, which requires setting δ to be exponentially small. Crucially, to maintain this low failure probability, the space of the offline coreset construction algorithm must also grow as $\text{poly}(\log \log \frac{1}{\delta})$, whereas most existing randomized algorithms grow as $\log \frac{1}{\delta}$. Therefore, the existing approach is mostly limited to using deterministic offline algorithms that tend to be slower than state-of-art

² For simplicity, the algorithm in Theorem 3 samples according to strong connectivity, which can be shown to give a lower bound on the size of the minimum cut intersecting e .



■ **Figure 1** Black-box wrapper: the sampler produces a compressed stream that is fed to merge-and-reduce, while merge-and-reduce supplies the coreset for computing sampling probabilities.

sampling-based routines. Our approach does not have this limitation, and can use any offline algorithm. The following theorem states the guarantees of our wrapper; the proof is provided in the full version of the paper.

► **Theorem 12.** *Let $\epsilon_1, \epsilon_2, \delta_1, \delta_2 > 0$. Suppose there exists an adversarially-robust self-weighted sampling algorithm $\mathcal{A}_{samp}$, that with probability $1 - \delta_1$, constructs a $(1 + \epsilon_1)$ -coreset of size $h(m, \epsilon_1, \delta_1)$, where m is the stream's length. Furthermore, suppose there exists an offline algorithm that computes with probability $1 - \delta_2$ a $(1 + \epsilon_2)$ -coreset of size $g(\epsilon_2, \delta_2)$.*

Then, for all $\epsilon, \delta > 0$, there exists an adversarially-robust algorithm that, given an adaptive stream of length m , with probability $1 - \delta$, outputs a $(1 + \epsilon)$ -coreset of size $O(g(\frac{\epsilon}{3 \log H}, \frac{\delta}{2H}) \cdot \log H)$, where $H := h(m, \frac{\epsilon}{3}, \frac{\delta}{2})$.

Hypergraph Cut Sparsification. We continue with the running example of hypergraph cut sparsification, and employ the reduction using the self-weighted online sampling algorithm of Theorem 3. The proof for subspace embeddings (Theorem 4) is analogous, and is deferred to the full version of the paper.

Proof of Theorem 2. We apply Theorem 12, with the adversarially-robust self-weighted online algorithm of Theorem 3, and an offline algorithm of size $K = \tilde{O}(\epsilon^{-2}n)$ that succeeds with probability $1 - 1/\text{poly}(n)$, e.g. [44]. Notice that as explained above, sampling probabilities rely only on the values of cuts in the hypergraph so far and hence can be computed using a sparsifier. By Theorem 3, the sparsity of w' is $m' = \tilde{O}(\epsilon^{-2}n^2 \log m)$. Plugging this into Theorem 12, we obtain the desired bound. Finally, the algorithm succeeds with probability $1 - 2^{-n} - 1/\text{poly}(n) = 1 - 1/\text{poly}(n)$. ◀

3 Importance Sampling with Adversarial Sensitivities

In this section, we prove Lemma 8, showing that for $a = O(\epsilon^{-2} \log \frac{\log \Delta}{\epsilon \delta})$, SamplingAlg wins the game against Adversary with probability $1 - \delta$. We will use the following definition and results concerning *martingales*.

► **Definition 13 (Martingale).** *A martingale is a sequence $X_0, X_1, \dots$ of random variables with finite mean, such that for every $i \geq 0$,*

$$\mathbb{E}[X_{i+1} | X_i, \dots, X_0] = X_i.$$

We use Freedman's inequality [28], which is an analogous version of Bernstein's inequality for martingales. Specifically, we use the following formulation, based on [47].

113:12 On the Adversarial Robustness of Online Importance Sampling

► **Theorem 14** (Freedman's Inequality). *Let $X_0, X_1, \dots, X_n$ be a martingale with $X_0 = 0$. Suppose there exists $M > 0, \sigma^2 > 0$ such that, for every $1 \leq i \leq n$, $|X_i - X_{i-1}| \leq M$ with probability 1 (a.s.), and the predictable quadratic variation satisfies*

$$\sum_{j=1}^i \text{Var}(X_j | X_{j-1}, \dots, X_0) \equiv \sum_{j=1}^i \mathbb{E}[(X_j - X_{j-1})^2 | X_{j-1}, \dots, X_0] \leq \sigma^2$$

with probability 1. Then, for every $\lambda > 0$,

$$\Pr(\max_{i \in [n]} |X_i| > \lambda) \leq 2 \exp\left(-\frac{\lambda^2/2}{\sigma^2 + M\lambda/3}\right).$$

We are now ready to prove Lemma 8.

Proof of Lemma 8. By Yao's principle, we can assume without loss of generality that Adversary is deterministic. That is, if there was a randomized adversary with randomness r that wins the game with probability $> \delta$, then there must be a choice for r for which the adversary wins with probability $> \delta$. Fixing r to this choice yields a deterministic adversary. Furthermore, note that we can assume that $x_1 = 1$ without loss of generality by rescaling.

Let m be the length of the stream. For every integer $0 \leq t \leq m$, let $X_t = \sum_{i=1}^t \tilde{x}_i - x_i$. We have $X_0 = 0$ and $X_t = X_{t-1} + \tilde{x}_t - x_t$ for $t \geq 1$, hence, $\mathbb{E}[X_t | X_{t-1}, \dots, X_0] = X_{t-1}$ and thus $X_0, X_1, \dots$ is a martingale. The difference sequence satisfies

$$|X_t - X_{t-1}| = |\tilde{x}_t - x_t| \leq \frac{1}{a} \sum_{i=1}^t x_i$$

and the variance satisfies

$$\text{Var}(X_t | X_{t-1}, \dots, X_0) = \frac{x_t^2}{p_t} - x_t^2 \leq \frac{x_t}{a} \sum_{i=1}^t x_i,$$

and thus the quadratic variation is $\sum_{i=1}^t \text{Var}(X_t | X_{t-1}, \dots, X_0) \leq \frac{1}{a} (\sum_{i=1}^t x_i)^2$.

We cannot use Freedman's inequality "as is", because $\sum_{i=1}^t x_i$ is a random variable. Instead, for the sake of analysis, we consider $L = O(\frac{1}{\epsilon} \log \Delta)$ stopped processes, as follows. For every $\ell \in [L]$, let τ_ℓ be the first time t for which $\sum_{i=1}^t x_i \geq (1+\epsilon)^\ell$. Since Adversary is deterministic, for every $t \leq m$, x_t is determined by $X_0, \dots, X_{t-1}$, hence it also determines the decision whether $t = \tau_\ell$ (i.e., τ_ℓ is a stopping time). We define $Y_{t,\ell}$ as the following random process: as long as $t \leq \tau_\ell - 1$, let $Y_{t,\ell} = X_t$. At $t = \tau_\ell$, let the residue be $R = (1+\epsilon)^\ell - \sum_{i=1}^{\tau_\ell-1} x_i$, and consider a virtual adversary, that inserts $x_{\tau_\ell,\ell} = R$ and $p_{\tau_\ell,\ell} = \min\{a \frac{R}{\sum_{i=1}^{\tau_\ell-1} x_i + R}, 1\}$. To simplify notations, denote by $\tilde{R}$ the response of SamplingAlg. Set $Y_{\tau_\ell,\ell} = X_{\tau_\ell-1} + \tilde{R} - R$, and for every $t > \tau_\ell$, $Y_{t,\ell} = Y_{t-1,\ell}$.

These random processes $Y_{t,\ell}$ are clearly still martingales, and their difference sequence and variance admit the following bounds. For $t < \tau_\ell$, the difference sequence satisfies

$$|Y_{t,\ell} - Y_{t-1,\ell}| \leq \frac{1}{a} \sum_{i=1}^t x_i \leq \frac{(1+\epsilon)^\ell}{a},$$

the variance satisfies

$$\text{Var}(Y_{t,\ell} | Y_{t-1,\ell}, \dots, Y_{0,\ell}) \leq \frac{x_t}{a} \sum_{i=1}^t x_i,$$

and hence, $\sum_{i=1}^t \text{Var}(Y_{t,\ell} | Y_{t-1,\ell}, \dots, Y_{0,\ell}) \leq \frac{(1+\epsilon)^{2\ell}}{a}$. These same bounds hold for $t = \tau_\ell$, and immediately also for $t > \tau_\ell$. By Freedman's inequality (Theorem 14),

$$\Pr[\max_{t \in [m]} |Y_{t,\ell}| > \epsilon(1+\epsilon)^\ell] \leq 2 \exp\left(-\frac{\epsilon^2(1+\epsilon)^{2\ell}/2}{\frac{1}{a}(1+\epsilon)^{2\ell} + \frac{\epsilon}{3a}(1+\epsilon)^{2\ell}}\right) \leq 2 \exp\left(-\frac{\epsilon^2 a}{3}\right).$$

For suitable $a = O(\epsilon^{-2} \log \frac{\log \Delta}{\epsilon \delta})$, the probability above is bounded by $\frac{\delta}{L}$. By a union bound, with probability at least $1 - \delta$, we have $\max_{t \in [m]} |Y_{t,\ell}| \leq \epsilon(1+\epsilon)^\ell$ for all $\ell \in [L]$.

In conclusion, for every $t \leq m$, we must have $\sum_{i=1}^t x_i \leq x_1 \Delta \leq \Delta$, where the last inequality is by our assumption that $x_1 = 1$, hence there exists $\ell \in [L]$ such that $\sum_{i=1}^t x_i \in [(1+\epsilon)^{\ell-1}, (1+\epsilon)^\ell]$. Therefore, $X_t = Y_{t,\ell}$, and we have

$$|X_t| \leq \max_{t \in [m]} |Y_{t,\ell}| \leq \epsilon(1+\epsilon)^\ell \leq \epsilon(1+\epsilon) \sum_{i=1}^t x_i.$$

Rescaling ϵ concludes the proof. ◀

References

- 1 Sara Ahmadian and Edith Cohen. Unmasking vulnerabilities: cardinality sketches under adaptive inputs. In *Proceedings of the 41st International Conference on Machine Learning, ICML. JMLR.org*, 2024.
- 2 Sara Ahmadian, Edith Cohen, and Uri Stemmer. The cost of compression: Tight quadratic black-box attacks on sketches for ℓ_2 norm estimation, 2025. To appear in NeurIPS '25. [arXiv:2507.16345](https://arxiv.org/abs/2507.16345).
- 3 Kook Jin Ahn and Sudipto Guha. Graph sparsification in the semi-streaming model. In *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009*, volume 5556 of *Lecture Notes in Computer Science*, pages 328–338. Springer, 2009. doi:10.1007/978-3-642-02930-1_27.
- 4 Miklós Ajtai, Vladimir Braverman, T. S. Jayram, Sandeep Silwal, Alec Sun, David P. Woodruff, and Samson Zhou. The white-box adversarial data stream model. In *PODS '22: International Conference on Management of Data*, pages 15–27. ACM, 2022. doi:10.1145/3517804.3526228.
- 5 Noga Alon, Omri Ben-Eliezer, Yuval Dagan, Shay Moran, Moni Naor, and Eylon Yogev. Adversarial laws of large numbers and optimal regret in online classification. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 447–455, 2021. doi:10.1145/3406325.3451041.
- 6 Sepehr Assadi, Amit Chakrabarti, Prantar Ghosh, and Manuel Stoeckl. Coloring in graph streams via deterministic and adversarially robust algorithms. In *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS*, pages 141–153, 2023. doi:10.1145/3584372.3588681.
- 7 Idan Attias, Edith Cohen, Moshe Shechner, and Uri Stemmer. A framework for adversarial streaming via differential privacy and difference estimators. *Algorithmica*, 86(11):3339–3394, 2024. doi:10.1007/S00453-024-01259-8.
- 8 Grey Ballard, Alex Druinsky, Nicholas Knight, and Oded Schwartz. Hypergraph partitioning for sparse matrix-matrix multiplication. *ACM Trans. Parallel Comput.*, 3(3):18:1–18:34, 2016. doi:10.1145/3015144.
- 9 Nikhil Bansal, Ola Svensson, and Luca Trevisan. New notions and constructions of sparsification for graphs and hypergraphs. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019*, pages 910–928. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00059.
- 10 Omri Ben-Eliezer, Talya Eden, and Krzysztof Onak. Adversarially robust streaming via dense-sparse trade-offs. In *5th Symposium on Simplicity in Algorithms, SOSA*, pages 214–227. SIAM, 2022. doi:10.1137/1.9781611977066.15.

- 11 Omri Ben-Eliezer, Rajesh Jayaram, David P. Woodruff, and Eylon Yogev. A framework for adversarially robust streaming algorithms. *J. ACM*, 69(2):17:1–17:33, 2022. doi:10.1145/3498334.
- 12 Omri Ben-Eliezer and Eylon Yogev. The adversarial robustness of sampling. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS*, pages 49–62, 2020. doi:10.1145/3375395.3387643.
- 13 András A. Benczúr and David R. Karger. Approximating s - t minimum cuts in $\tilde{O}(n^2)$ time. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 47–55. ACM, 1996. doi:10.1145/237814.237827.
- 14 Daniel Berend and Tamir Tassa. Improved bounds on bell numbers and on moments of sums of random variables. *Probability and Mathematical Statistics*, 30(2):185–205, 2010.
- 15 Vladimir Braverman, Petros Drineas, Cameron Musco, Christopher Musco, Jalaj Upadhyay, David P. Woodruff, and Samson Zhou. Near optimal linear algebra in the online and sliding window models. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 517–528, 2020. doi:10.1109/FOCS46700.2020.00055.
- 16 Vladimir Braverman, Avinatan Hassidim, Yossi Matias, Mariano Schain, Sandeep Silwal, and Samson Zhou. Adversarial robustness of streaming algorithms through importance sampling. In *Advances in Neural Information Processing Systems, NeurIPS*, pages 3544–3557, 2021. URL: <https://proceedings.neurips.cc/paper/2021/hash/1d01bd2e16f57892f0954902899f0692-Abstract.html>.
- 17 Amit Chakrabarti, Prantar Ghosh, and Manuel Stoeckl. Adversarially robust coloring for graph streams. In *13th Innovations in Theoretical Computer Science Conference, ITCS*, volume 215 of *LIPIcs*, pages 37:1–37:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ITCS.2022.37.
- 18 Amit Chakrabarti and Manuel Stoeckl. Finding missing items requires strong forms of randomness. In *39th Computational Complexity Conference, CCC*, volume 300 of *LIPIcs*, pages 28:1–28:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.CCC.2024.28.
- 19 Chandra Chekuri and Chao Xu. Minimum cuts and sparsification in hypergraphs. *SIAM J. Comput.*, 47(6):2118–2156, 2018. doi:10.1137/18M1163865.
- 20 Yu Chen, Sanjeev Khanna, and Ansh Nagda. Sublinear time hypergraph sparsification via cut and edge sampling queries. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021*, volume 198 of *LIPIcs*, pages 53:1–53:21, 2021. doi:10.4230/LIPIcs.ICALP.2021.53.
- 21 Edith Cohen, Xin Lyu, Jelani Nelson, Tamas Sarlos, Moshe Shechner, and Uri Stemmer. On the robustness of CountSketch to adaptive inputs. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 4112–4140. PMLR, 2022. URL: <https://proceedings.mlr.press/v162/cohen22a.html>.
- 22 Edith Cohen, Jelani Nelson, Tamás Sarlós, and Uri Stemmer. Tricking the hashing trick: a tight lower bound on the robustness of counts sketch to adaptive inputs. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence*. AAAI Press, 2023. doi:10.1609/aaai.v37i6.25882.
- 23 Edith Cohen, Jelani Nelson, Tamás Sarlós, Mihir Singhal, and Uri Stemmer. One attack to rule them all: Tight quadratic bounds for adaptive queries on cardinality sketches, 2025. To appear in SODA '26. arXiv:2411.06370.
- 24 Michael B. Cohen, Cameron Musco, and Jakub Pachocki. Online row sampling. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016*, volume 60 of *LIPIcs*, pages 7:1–7:18, 2016. doi:10.4230/LIPIcs.APPROX-RANDOM.2016.7.
- 25 Michael B. Cohen and Richard Peng. L_p row sampling by lewis weights. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC*, pages 183–192, 2015. doi:10.1145/2746539.2746567.

- 26 Vincent Cohen-Addad, David P. Woodruff, Shenghao Xie, and Samson Zhou. Nearly space-optimal graph and hypergraph sparsification in insertion-only data streams, 2025. doi:10.48550/arXiv.2510.18180.
- 27 Vincent Cohen-Addad, David P. Woodruff, and Samson Zhou. Streaming euclidean k-median and k-means with $o(\log n)$ space. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 883–908, 2023. doi:10.1109/FOCS57990.2023.00057.
- 28 David A. Freedman. On Tail Probabilities for Martingales. *The Annals of Probability*, 3(1):100–118, 1975. doi:10.1214/aop/1176996452.
- 29 Elena Gribelyuk, Honghao Lin, David P. Woodruff, Huacheng Yu, and Samson Zhou. A strong separation for adversarially robust ℓ_0 estimation for linear sketches. In *IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2318–2343, 2024. doi:10.1109/FOCS61266.2024.00136.
- 30 Elena Gribelyuk, Honghao Lin, David P. Woodruff, Huacheng Yu, and Samson Zhou. Lifting linear sketches: Optimal bounds and adversarial robustness. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, pages 395–406. Association for Computing Machinery, 2025. doi:10.1145/3717823.3718227.
- 31 Sudipto Guha, Andrew McGregor, and David Tench. Vertex and hyperedge connectivity in dynamic graph streams. In *Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015*, pages 241–247. ACM, 2015. doi:10.1145/2745754.2745763.
- 32 Moritz Hardt and David P. Woodruff. How robust are linear sketches to adaptive inputs? In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, pages 121–130. Association for Computing Machinery, 2013. doi:10.1145/2488608.2488624.
- 33 Avinatan Hassidim, Haim Kaplan, Yishay Mansour, Yossi Matias, and Uri Stemmer. Adversarially robust streaming algorithms via differential privacy. *J. ACM*, 69(6):42:1–42:14, 2022. doi:10.1145/3556972.
- 34 Shunhua Jiang, Binghui Peng, and Omri Weinstein. The complexity of dynamic least-squares regression. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023*, pages 1605–1627. IEEE, 2023. doi:10.1109/FOCS57990.2023.00097.
- 35 Haim Kaplan, Yishay Mansour, Kobbi Nissim, and Uri Stemmer. Separating adaptive streaming from oblivious streaming using the bounded storage model. In *Advances in Cryptology – CRYPTO 2021 – 41st Annual International Cryptology Conference, CRYPTO*, volume 12827 of *Lecture Notes in Computer Science*, pages 94–121. Springer, 2021. doi:10.1007/978-3-030-84252-9_4.
- 36 Michael Kapralov, Robert Krauthgamer, Jakab Tardos, and Yuichi Yoshida. Towards tight bounds for spectral sparsification of hypergraphs. In *STOC '21*, pages 598–611. ACM, 2021. doi:10.1145/3406325.3451061.
- 37 Sanjeev Khanna, Huan Li, and Aaron Putterman. Near-optimal linear sketches and fully-dynamic algorithms for hypergraph spectral sparsification. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025*, pages 1190–1200. ACM, 2025. doi:10.1145/3717823.3718239.
- 38 Sanjeev Khanna, Aaron Putterman, and Madhu Sudan. Near-optimal size linear sketches for hypergraph cut sparsifiers. In *65th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2024*, pages 1669–1706. IEEE, 2024. doi:10.1109/FOCS61266.2024.00105.
- 39 Sanjeev Khanna, Aaron Putterman, and Madhu Sudan. Near-optimal hypergraph sparsification in insertion-only and bounded-deletion streams. In *52nd International Colloquium on Automata, Languages, and Programming, ICALP 2025*, volume 334 of *LIPIcs*, pages 108:1–108:11. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.ICALP.2025.108.
- 40 Dmitry Kogan and Robert Krauthgamer. Sketching cuts in graphs and hypergraphs. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015*, pages 367–376. ACM, 2015. doi:10.1145/2688073.2688093.

- 41 Yi Li and Mingmou Liu. Lower bounds for sparse oblivious subspace embeddings. In *PODS '22: International Conference on Management of Data*, pages 251–260. ACM, 2022. doi:10.1145/3517804.3526224.
- 42 Per-Gunnar Martinsson and Joel A. Tropp. Randomized numerical linear algebra: Foundations and algorithms. *Acta Numer.*, 29:403–572, 2020. doi:10.1017/S0962492920000021.
- 43 Cameron Musco, Christopher Musco, David P. Woodruff, and Taisuke Yasuda. Active linear regression for ℓ_p norms and beyond. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 744–753, 2022. doi:10.1109/FOCS54457.2022.00076.
- 44 Kent Quanrud. Quotient sparsification for submodular functions. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024*. SIAM, 2024. doi:10.1137/1.9781611977912.187.
- 45 Tasuku Soma, Kam Chuen Tung, and Yuichi Yoshida. Online algorithms for spectral hypergraph sparsification. In *Integer Programming and Combinatorial Optimization – 25th International Conference, IPCO 2024*, volume 14679 of *Lecture Notes in Computer Science*, pages 405–417. Springer, 2024. doi:10.1007/978-3-031-59835-7_30.
- 46 Manuel Stoeckl. Streaming algorithms for the missing item finding problem. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 793–818, 2023. doi:10.1137/1.9781611977554.CH32.
- 47 Joel Tropp. Freedman’s inequality for matrix martingales. *Electronic Communications in Probability*, 16(none):262–270, 2011. doi:10.1214/ECP.v16-1624.
- 48 David P. Woodruff. Sketching as a tool for numerical linear algebra. *Found. Trends Theor. Comput. Sci.*, 10(1-2):1–157, 2014. doi:10.1561/04000000060.
- 49 David P. Woodruff and Taisuke Yasuda. Online lewis weight sampling. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 4622–4666, 2023. doi:10.1137/1.9781611977554.CH175.
- 50 David P. Woodruff and Samson Zhou. Tight bounds for adversarially robust streams and sliding windows via difference estimators. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 1183–1196, 2021. doi:10.1109/FOCS52979.2021.00116.
- 51 David P. Woodruff and Samson Zhou. Adversarially robust dense-sparse tradeoffs via heavy-hitters. In *Advances in Neural Information Processing Systems NeurIPS*, 2024. URL: https://papers.nips.cc/paper_files/paper/2024/hash/14c00f4bc19a5498982b16647998e894-Abstract-Conference.html.
- 52 Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha P. Talukdar. Hypercnn: A new method for training graph convolutional networks on hypergraphs. In *Advances in Neural Information Processing Systems, NeurIPS*, pages 1509–1520, 2019. URL: <https://proceedings.neurips.cc/paper/2019/hash/1efa39bcaec6f3900149160693694536-Abstract.html>.
- 53 Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1601–1608. MIT Press, 2006. URL: <https://proceedings.neurips.cc/paper/2006/hash/df8e9c2ac33381546d96deea9922999-Abstract.html>.

A Application: Unweighted Hypergraph Cut Sparsification

This section proves Theorem 3. It is similar to the construction of cut sparsifiers for graphs using online sampling provided in [3].

We begin by presenting several important definitions, which are based on the work of [44, 38]. Let $H = (V, E)$ be an unweighted hypergraph. For every partition $V_1, \dots, V_k$ of V , let $E[V_1, \dots, V_k]$ denote the set of hyperedges that are not entirely contained in any of the V_i ’s. The structural properties of hypergraphs which allow us to bound the size of the sparsifier rely on the notion of *normalized cuts*. For every $k \in [2, |V|]$, a k -cut in H is

a partition of the vertex set V into k disjoint sets $V_1, \dots, V_k$. The value of the cut is the number of hyperedges that intersect the cut, denoted by $\text{cut}_H(V_1, \dots, V_k) := |E[V_1, \dots, V_k]|$. Finally, the *normalized cut value* of a k -cut is defined as $|E[V_1, \dots, V_k]|/(k-1)$, we denote the minimum normalized cut value of H by $\lambda(H)$.

For every vertex subset $W \subseteq V$, let $H[W]$ be the sub-hypergraph of H induced by W , i.e. the hypergraph on the vertices W that includes only hyperedges $e \in E$ such that $e \subseteq W$. The *strength* of a hyperedge $e \in E$ is given by

$$\kappa_e^H = \max_{W \subseteq V} \lambda(H[W \cup e]),$$

where we remove the superscript H when it is clear from context. We will also need the following fact.

► **Fact 15.** *Let n be an integer. Summing over all $k \in [2, n]$, the number of k -cuts in a hypergraph on n vertices is the bell number B_n , which in turn is bounded by (Theorem 3.1 from [14]),*

$$B_n < \left(\frac{0.792n}{\log(n+1)} \right)^n \leq 2^{n \cdot \log n}.$$

A.1 Proof of Theorem 3

Note that we prove the theorem for the stronger notion of k -cut sparsifiers, which preserve all k -cuts for $k \in [2, n]$ up to multiplicative $(1 \pm \epsilon)$ factor. The algorithm used for constructing the sparsifier is presented in Algorithm 1. We prove Theorem 3 by showing that the algorithm returns a small $(1 \pm \epsilon)$ -cut sparsifier of the hypergraph H with high probability. The proof of the theorem is split into two parts: 1) Showing that the output of the algorithm is a $(1 \pm \epsilon)$ cut sparsifier with high probability, and 2) bounding the number of hyperedges in the resulting sparsifier.

For every $i \in [m]$, let $H_i = (V, E_i = \{e_1, \dots, e_i\})$ be the hypergraph on the first i hyperedges, and let $H'_i = (V, E'_i, w')$ be the sparsifier after the i -th insertion, note that H'_i is a weighted hypergraph with weight function $w' : E'_i \rightarrow \mathbb{R}_{>0}$.

► **Lemma 16 (Correctness).** *For every adaptive adversary and $i \in [m]$, with probability at least $1 - 2^{-4n}$, Algorithm 1 outputs a $(1 \pm \epsilon)$ -cut sparsifier H'_i of H_i .*

► **Lemma 17 (Size).** *The number of hyperedges in the output of Algorithm 1 is $O(\epsilon^{-2} n^2 \log m)$ with probability at least $1 - 2^{-4n}$.*

Theorem 3 follows by a union bound on the two events.

Algorithm 1 SAMPLE-HYPERGRAPH.

```

1:  $H' \leftarrow (V, E' = \emptyset)$ 
2:  $\rho \leftarrow K_1 \epsilon^{-2} n \log n$  ▷ where  $K_1$  is a large enough constant
3: while new edge  $e_i$  do
4:    $\text{coin} \leftarrow \text{True}$  with probability  $p_i = \min\{\rho/\kappa_{e_i}^{H'_i}, 1\}$ , and otherwise  $\text{coin} \leftarrow \text{False}$ 
5:   if  $\text{coin}$  then
6:      $E' \leftarrow E' \cup \{e_i\}$ 
7:      $w'_{e_i} \leftarrow \frac{1}{p_i}$ 
8:   output  $\text{coin}$  ▷ may also output  $H'$ 

```

Proof of Lemma 16. Fix a k -cut $(V_1^*, \dots, V_k^*)$ and consider a hyperedge e_i that intersects the cut. Observe that since the cut intersects the hyperedge e_i , it separates the κ_{e_i} -strong component W containing e . Let $W_1, \dots, W_{k'}$ be the partition of W induced by the cut $(V_1^*, \dots, V_k^*)$, where $k' \leq k$. By definition, we have $\kappa_{e_i} \leq \text{cut}_{H'_i[W]}(W_1, \dots, W_{k'}) / (k' - 1) \leq \text{cut}_{H'_i[W]}(W_1, \dots, W_k)$ and since expanding the cut to the entire hypergraph H'_i does not decrease the cut value, we have $\kappa_{e_i} \leq \text{cut}_{H'_i}(V_1^*, \dots, V_k^*)$. Therefore, the sampling probability satisfies $p_{e_i} = \min\{\rho / \kappa_{e_i}, 1\} \geq \min\{\rho / \text{cut}_{H'_i}(V_1^*, \dots, V_k^*), 1\}$.

This is precisely the setting of Theorem 7, since the maximum value of each cut is at most m and its minimum value is at least 1. Recalling that $T = m \leq 2^n$, $\delta = 2^{-5n \log n}$ and setting $\rho = O(\epsilon^{-2} \log \frac{\log T}{\epsilon \delta}) = O(\epsilon^{-2} n \log n)$, the probability that the cut is preserved is at least $1 - 2^{-5n \log n}$. The proof concludes by applying a union bound over all $2^{n \log n}$ k -cuts. $\blacktriangleleft$

We now turn to bound the number of hyperedges in the sparsifier, proving Lemma 17. The proof is similar to Theorem 3.2 in [3].

Proof of Lemma 17. We begin by proving several useful claims about hyperedge strengths. The first claim is an extension of [13, Lemma 3.1], on the occurrence of α -strong components, to hypergraphs. Recall that a component $A \subseteq V$ is called α -strong if every normalized k -cut $A_1, \dots, A_k$ of A satisfies $\text{cut}_H(A_1, \dots, A_k) / (k - 1) \geq \alpha$.

$\triangleright$ **Claim 18.** A hypergraph with total hyperedge weight at least $\alpha \cdot (n - 1)$ has an α -strong component.

Proof. The proof is by contradiction. Let n be the minimum integer for which there exists a counter example, i.e., a weighted hypergraph $G = (V, E, w)$ that has total hyperedge weight at least $\alpha \cdot (n - 1)$, but no α -strong component. In particular, G is not α -strong. Hence, there exists a k -cut, $V_1, \dots, V_k$ in G with normalized cut value at most $\text{cut}_G(V_1, \dots, V_k) / (k - 1) < \alpha$, for some $k \leq n$.

Denote $n_i = |V_i|$ and for every vertex set $S \subseteq V$, denote by $E[S]$ the set of hyperedges in the induced hypergraph $G[S]$. By the minimality of n , the total hyperedge weight in $G[V_i]$ is at most $\alpha \cdot (n_i - 1)$ for all $i \in [k]$. Therefore, summing the total weight of hyperedges,

$$\text{cut}_G(V_1, \dots, V_k) + \sum_{i=1}^k w(E[V_i]) < \alpha(k - 1) + \sum_{i=1}^k \alpha(n_i - 1) = \alpha(n - 1),$$

which is in contradiction to the total weight of the hyperedges in G . Therefore, no such counter example exists. $\triangleleft$

Next we prove the following useful claim bounding the total weight of hyperedges in the sparsifier.

$\triangleright$ **Claim 19.** If H'_i is a $(1 \pm \epsilon)$ cut sparsifier of H_i then $\sum_{e \in E'_i} w'_e \leq (1 + \epsilon)n/2 \cdot |E_i|$.

Proof. Observe that

$$\sum_{v \in V} \text{cut}_{H_i}(\{v\}, V \setminus \{v\}) \leq n \cdot |E_i|,$$

since every hyperedge is counted at most n times. Similarly, we have that

$$2 \cdot \sum_{e \in E'_i} w'_e \leq \sum_{v \in V} \text{cut}_{H'_i}(\{v\}, V \setminus \{v\}) \leq (1 + \epsilon) \sum_{v \in V} \text{cut}_{H_i}(\{v\}, V \setminus \{v\}) \leq (1 + \epsilon)n \cdot |E_i|,$$

where the first inequality is since every hyperedge is counted at least twice. $\triangleleft$

Let $F_\kappa = \{e_j \in E'_i \mid j \leq i, \kappa_{e_j} \leq \kappa\}$, be the set of all sampled hyperedges that had strength at most κ in $H'_{j-1} \cup e_j$ when they were added. The following claim bounds the total weight of hyperedges in F_κ .

▷ **Claim 20.** The total weight of hyperedges in F_κ is at most $n\kappa(1 + 1/\rho)$.

Proof. Let $G_\kappa = (V, F_\kappa)$ be the sub-hypergraph of the sparsifier that comprises of all the hyperedges in F_κ . Observe that if G_κ has no $(\kappa + \kappa/\rho + 1)$ -strong component then the total weight of hyperedges in F_κ is at most $n(\kappa + \kappa/\rho)$ by Claim 18. Therefore, assume towards contradiction that G_κ has a $(\kappa + \kappa/\rho + 1)$ -strong component.

Let e be the first edge that was sampled into F_κ that is in the $(\kappa + \kappa/\rho + 1)$ -strong component. Notice that since H is an unweighted hypergraph, the weight of e in the sparsifier is at most $p_e^{-1} \leq \rho/\kappa$. Hence, removing e can decrease the strength of the component by at most κ/ρ ; therefore $G_\kappa \setminus e$ has a $(\kappa + 1)$ -strong component in contradiction to e being sampled with strength at most κ . ◁

We now bound the number of hyperedges in the sparsifier. Assume that H'_i is a $(1 + \epsilon)$ cut sparsifier of H_i , this holds for all i with probability at least $1 - 2^{-4n}$ by Lemma 16. By Claim 19, $\sum_{e \in E'_i} w'_e \leq (1 + \epsilon) \cdot n|E_i|/2$. Thus, for all $e' \in E'_i$, $\kappa_{e'} \leq (1 + \epsilon)\rho n \cdot |E_i|/2$, since the maximum cut in the graph is $\leq \sum_{e \in E'_i} w'_e$ which is also an upper bound on $\kappa_{e'}$. Denote this upper bound on κ by κ^* . Therefore, the number of hyperedges is bounded by

$$\begin{aligned}
|E'| &= \sum_{\kappa=1}^{\kappa^*} |F_\kappa \setminus F_{\kappa-1}| \leq \sum_{\kappa=1}^{\kappa^*} \frac{\rho}{\kappa} \sum_{e \in F_\kappa \setminus F_{\kappa-1}} w'_e && \text{since } w'_e = \frac{1}{p_e} \geq \frac{\kappa_e}{\rho} \\
&= \sum_{\kappa=1}^{\kappa^*} \frac{\rho}{\kappa} (w'(F_\kappa) - w'(F_{\kappa-1})) = \sum_{\kappa=1}^{\kappa^*} \frac{\rho}{\kappa} w'(F_\kappa) - \sum_{\kappa=0}^{\kappa^*-1} \frac{\rho}{\kappa+1} w'(F_\kappa) \\
&= \frac{\rho}{\kappa^*+1} w'(F_{\kappa^*}) + \rho \sum_{\kappa=1}^{\kappa^*} \left(\frac{1}{\kappa} - \frac{1}{\kappa+1} \right) w'(F_\kappa) && \text{since } F_0 = \emptyset \\
&\leq \rho n \left(1 + \frac{1}{\rho}\right) + \rho \sum_{\kappa=1}^{\kappa^*} \frac{1}{\kappa+1} n \left(1 + \frac{1}{\rho}\right) && \text{by Claim 20} \\
&= O(\rho n \log \kappa^*) = O(\epsilon^{-2} n^2 \log m).
\end{aligned}$$

This concludes the proof of Lemma 17. ◀

A.2 Coreset Properties

In this section we show that cut sparsifiers satisfy the properties of coresets that were presented in Section 2. The properties are the following, merge, reduce and linear cost function. We begin by showing that they satisfy the merge and reduce properties.

Let E_1, E_2 be two sets of hyperedges over a vertex set V , $H_i = (V, E_i)$ and H'_i a quality $(1 \pm \epsilon_i)$ -cut sparsifier for H_i , for some $\epsilon_i \in (0, 1)$. It is easy to see that $\text{cut}_{(V, E_1 \cup E_2)}(S) = \text{cut}_{H_1}(S) + \text{cut}_{H_2}(S)$. Furthermore, $\text{cut}_{H'_1}(S) + \text{cut}_{H'_2}(S) \in (1 \pm \max\{\epsilon_1, \epsilon_2\}) \cdot \text{cut}_{(V, E_1 \cup E_2)}(S)$. Therefore, the merge property holds.

To prove the reduce property, let H be some hypergraph, H' be a quality $(1 \pm \epsilon_1)$ sparsifier of H and H'' be a quality $(1 \pm \epsilon_2)$ sparsifier of H' . Notice that,

$$\forall S \subseteq V, \text{cut}_{H''}(S) \in (1 \pm \epsilon_2) \text{cut}_{H'}(S) \in (1 \pm \epsilon_2)(1 \pm \epsilon_1) \text{cut}_H(S)$$

Finally, notice that the cost function $c : \mathcal{U} \times \mathcal{Q} \rightarrow \mathbb{R}_+$ is simply $c(e, U) = \mathbb{1}_{\{0 < |e \cap U| < |e|\}}$, and hence the cost function C satisfies the conditions of Definition 9.